# O Manual do Kturtle

Cies Breijs <cies AT kde DOT nl>
Anne-Marie Mahfouf <annma AT kde DOT org>
Tradução: Marcus Gama
Revisão 0.6 (2004-11-8)


O KTurtle é um ambiente de programação educativo que usa a linguagem de programação Logo. A qualidade única do LOGO é que os comandos de programação são traduzidos para o idioma do 'programador', para que ele possa programar no seu idioma nativo.

Índice

Lista de Tabelas

# Capítulo 1.

## Introdução

O KTurtle é um ambiente educativo de programação utilizando a linguagem de programação Logo. Ele tenta manter a programação tão acessível quanto possível. Isto torna o KTurtle adequado para ensinar às crianças matemática, geometria e... programação. Os comandos usados para programar são do estilo da linguagem de programação Logo. Um recurso único do Logo é que os comandos são normalmente traduzidos para o idioma falado pelo programador.

O KTurtle tem o nome com base na “tartaruga” que desempenha um papel central no ambiente de programação. O usuário programa a tartaruga, usando os comandos do Logo, para desenhar uma imagem na área de desenho.

## O que é o Logo?

A primeira versão da linguagem de programação Logo foi criada por Seymour Papert do Laboratório de Inteligência Artificial do MIT em 1967 como uma alternativa à linguagem de programação LISP. Desde então, foram lançadas várias versões do Logo. Em 1980, o Logo foi ganhando adeptos, com versões para o MSX, Commodore, Atari e sistemas IBM PC. Estas versões eram principalmente para fins educacionais. A LCSI lançou o MacLogo em 1985 como uma ferramenta para programadores profissionais, mas nunca teve grande sucesso. O MIT ainda mantém um sítio sobre Logo que poderá ser acessado em: http://el.media.mit.edu/logo-foundation/.

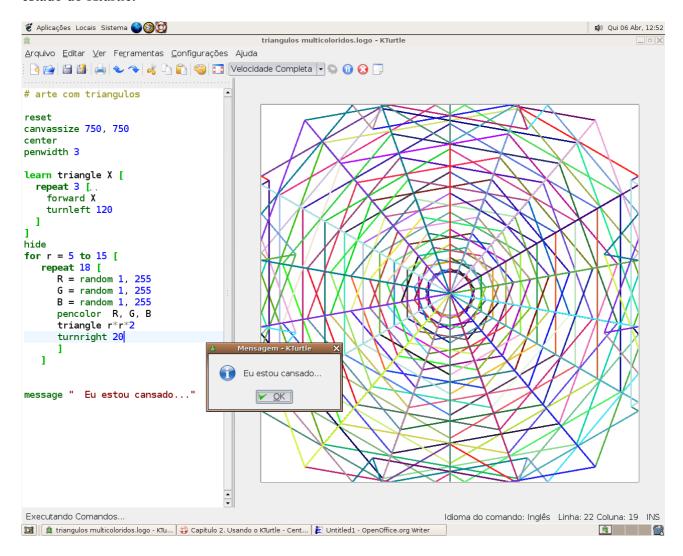
Hoje em dia existem várias versões do Logo por aí, que poderão ser encontradas no sítio de Logo do MIT e com uma pequena pesquisa no Google. Esta versão do Logo (KTurtle) é focada somente nas qualidades educacionais da linguagem de programação e não tentará se adequar às necessidades profissionais dos programadores.

## Recursos do KTurtle

O KTurtle possui alguns recursos legais que tornam a introdução à programação uma leve brisa. Veja aqui alguns dos detalhes dos recursos do KTurtle:

- Um interpretador de Logo integrado (sem dependências extras) que usa arquivos em XML para as traduções dos comandos, suporta as funções definidas pelo usuário e a mudança dinâmica de tipos.
- A execução pode ser tornada mais lenta, pausada ou interrompida a qualquer momento.
- Um editor poderoso para os comandos de Logo com um realce de sintaxe intuitivo, com numeração de linhas, entre outras coisas.
- A área de desenho pode ser gravada como uma imagem ou impressa.
- A área de desenho tem um modo de envolvência (ativo por padrão), para que a tartaruga não se perca assim tão facilmente.
- A ajuda de contexto para todos os comandos do Logo, basta para isso pressionar F2.
- Os comandos de Logo são completamente traduzíveis (atualmente só é suportado o Inglês, o Holandês e o Francês, o Esloveno, o Sérvio (Cirílico e Latim), Espanhol e Sueco).
- Um diálogo de erros que associa as mensagens de erro aos erros propriamente ditos no programa.
- Uma terminologia de programação simplificada.
- Modo de tela cheia.
- Vários programas exemplo em Logo integrados facilitam o início.

# Capítulo 2. Usando o Kturtle

A janela principal do KTurtle tem dois componentes principais, o editor de código (3) à esquerda, onde você escreve os comandos de Logo e a área de desenho (4) à direita, onde as instruções são visualizadas. A área de desenho é a área de recreio da tartaruga; é na área de desenho que a tartaruga se move e desenha. Os três outros locais da janela principal são: o menu (1) onde todas as ações podem ser acessadas, a barra de ferramentas (4) que lhe permite selecionar rapidamente as ações mais utilizadas e a barra de estado (5) onde você irá encontrar algumas informações sobre o estado do Kturtle.

# O Editor de Código

No editor de código, você poderá escrever os comandos de Logo. Ele tem todos os recursos que você poderia esperar num editor moderno. A maioria dos seus recursos são encontrados nos menus Editar e Ferramentas. O editor de código pode ser acoplado a qualquer um dos lados da janela principal ou poderá ser destacado e colocado em qualquer local da sua tela.

Você dispõe de várias formas de obter algum código no editor. A forma mais fácil é usar um exemplo já feito. Você escolhe o Arquivo->Abrir Exemplos no menu Arquivo, onde poderá clicar num arquivo. O nome do arquivo lhe dirá algo sobre o que é o exemplo (por exemplo, o 'square.logo' vai desenhar um quadrado ('square'). O arquivo que escolher será aberto no editor de código, você poderá ir então a Arquivo->Executar Comandos para rodar o código se desejar.

Você poderá abrir os arquivos de Logo escolhendo Arquivo->Abrir.

A terceira forma é escrever diretamente o seu código no editor ou copiar/colar algum código deste guia do usuário.

A posição do cursor é indicada na barra de estado à direita, com o número da Linha e da Coluna.

## A Área de Desenho

A área de desenho é a área onde os comandos são visualizados, ou seja, onde eles “desenham” uma imagem. Em outras palavras, é o espaço de recreio da tartaruga. Depois de introduzir algum código no editor de código e de executá-lo com Arquivo->Executar Comandos, duas coisas poderão acontecer: ou o código se executa com perfeição e você poderá ver algo mudando na área de desenho, ou você tem um erro no seu código e existirá uma mensagem que lhe dirá qual o erro que você cometeu.

Esta mensagem deverá ajudá-lo a resolver o erro.

A imagem que é desenhada pode ser gravada num arquivo (usando o Arquivo->Salvar a Área de Desenho) ou impressa (usando o Arquivo->Imprimir...).

## A Barra de Menu

No menu, você irá encontrar todas as ações do KTurtle. Elas estão nos seguintes grupos: Arquivo, Editar, Ver, Ferramentas, Configurações e Ajuda. Esta seção descreve todas estas opções.

As ações do Menu Arquivo:

Arquivo->Novo (Ctrl-N): Cria um arquivo de Logo novo, vazio.

Arquivo->Abrir... (Ctrl-O): Abre um arquivo de Logo.

Arquivo->Abrir Recente: Abre um arquivo de Logo que foi aberto recentemente.

Arquivo->Abrir Exemplos (Ctrl-E): Mostra a pasta com os programas de exemplo do Logo. Os exemplos deverão estar no seu idioma favorito, que poderá ser escolhido em Configurações->Configurar o KTurtle....

Arquivo->Executar Comandos (Alt-Return) : Inicia a execução dos comandos no editor de código.

Arquivo->Salvar (Ctrl-S): Salva o arquivo de Logo aberto atualmente.

Arquivo->Salvar Como...: Salva o arquivo de Logo aberto atualmente num local especificado.

Arquivo->Salvar Área de Desenho: Salva a imagem desenhada na área de desenho num arquivo.

Arquivo->Pausar a Execução (Pause) : Coloca a execução em pausa. Esta ação só fica ativa quando os comandos estiverem de fato sendo executados.

Arquivo->Parar Execução (Escape): Pára a execução; esta ação só fica ativa quando os comandos estiverem de fato sendo executados.

Arquivo->Velocidade de Execução: Apresenta uma lista com as velocidades de execução possíveis, consistindo em: Toda Velocidade, Lento, Mais Lento e Lentíssimo. Quando a velocidade é igual a 'Toda Velocidade' (padrão), você poderá não conseguir ver o que está acontecendo. Em alguns dos casos este comportamento poderá ser o desejado, mas noutros casos você poderá querer ter uma idéia da execução. No último caso, poderá desejar configurar a velocidade da execução como 'Lento', 'Mais Lento' e 'Lentíssimo'. Quando um dos modos lentos for selecionado, a posição atual de execução será mostrada no editor.

Arquivo->Imprimir... (Ctrl-P): Imprime o código atual no editor ou então o desenho gerado na área de desenho.

Arquivo->Sair (Ctrl-Q): Sai do Kturtle.

No Menu Editar temos:

Editar->Desfazer (Ctrl-Z): Anula a última alteração ao código. O KTurtle pode fazer anulações de forma ilimitada.
Editar->Refazer (Ctrl-Shift-Z): Refaz uma alteração anulada ao código.
Editar->Cortar (Ctrl-X): Corta o texto selecionado do editor de código para a área de transferência.
Editar->Copiar (Ctrl-C): Copia o texto selecionado do editor de código para a área de transferência.
Editar->Colar (Ctrl-V): Cola o texto na área de transferência no editor.
Editar->Procurar... (Ctrl-F): Com esta ação, você poderá procurar frases no código.
Editar->Procurar Próximo (F3): Use isto para procurar a próxima ocorrência da frase.
Editar->Substituir... (Ctrl-R): Com esta ação, você poderá substituir frases no código.

No Menu Ver podemos:

Ver->Modo Tela Cheia (Ctrl-Shift-F): Com esta ação, você ativa ou desativa o modo de tela cheia.
Nota: Quando o código é executado, estando no modo de tela cheia, tudo exceto a área de desenho fica escondido. Isto torna possível criar programas de "tela cheia" no KTurtle.
Ver->Mostrar Números de Linha (F11): Com esta ação, você poderá mostrar os números de linha no editor de código. Isto poderá ser útil para procurar um erro.

Opções do Menu Ferramentas:

Ferramentas->Extrator de Cor (Alt-C): Esta ação abre o extrator de cores. Com este extrator, você poderá selecionar facilmente um código de cores e inserí-lo no editor de código.
Ferramentas->Indentar (Ctrl-I): Esta ação "indenta" (adiciona espaços em branco) as linhas que estão selecionadas. Quando a 'indentação' é usada convenientemente, isto poderá tornar o código muito mais fácil de ler. Todos os exemplos usam indentação, por isso veja-os, por favor.
Ferramentas->Desindentar (Ctrl-Shift-I): Esta ação "desindenta" (remove os espaços em branco no início) as linhas que estão selecionadas.
Ferramentas->Limpar Indentação: Esta ação limpa a "indentação" (remove todos os espaços em branco no início) das linhas que estão selecionadas.
Ferramentas->Comentar (Ctrl-D): Esta ação adiciona caracteres de comentário (#) ao início das linhas que são selecionadas. As linhas que começam com um carácter de comentário são ignoradas quando o código é executado. Os comentários permitem ao programador explicar um pouco o seu código ou podem ser usadas para evitar temporariamente que um determinado pedaço de código seja executado.
Ferramentas->Descomentar (Ctrl-Shift-D): Esta ação remove os caracteres de comentários das linhas selecionadas.

O Menu Configurações:

Configurações->Mostrar/Ocultar Barra de Ferramentas: Ativa ou desativa a Barra Principal
Configurações->Mostrar/Ocultar Barra de Estado: Alterna a Barra de Estado
Configurações->Configurações Avançadas: Aqui você poderá alterar as coisas que você normalmente não necessita mudar. O sub-menu da Configurações Avançadas possui três itens: Configurar o Editor... (a janela de configuração padrão do editor Kate), Configurar os Atalhos... (a janela de configuração de atalhos padrão do KDE) e a Configurar as Barras de Ferramentas... (a janela de configuração da barra de ferramentas do KDE).
Configurações->Configurar o Kturtle...: Isto é usado para configurar o KTurtle. Aqui você poderá mudar o idioma dos comandos de Logo ou definir um novo tamanho inicial para a área de desenho.

O Menu Ajuda:

Ajuda->Manual do Kturtle: Esta ação mostra o manual que você está atualmente lendo.
Ajuda->O Que É Isto? (Shift-F1): Depois de ativar esta ação, a seta do cursor irá mudar para uma “seta com ponto de interrogação”. Este novo tipo de seta é usado para clicar em partes da janela principal do KTurtle, aparecerá uma descrição da componente em particular.
Ajuda->Ajuda sobre: ... (F1): Esta é uma função bastante útil; ela fornece ajuda sobre o código sobre o qual o cursor se encontra. Por isso, por exemplo, você poderá usar o comando print no seu código, e poderá querer ler para saber o que o manual diz sobre este comando. Você poderá mover o cursor para cima do comando print e clicar em F1. O manual irá então mostrar toda a informação sobre o comando print.
Esta função é muito importante durante a aprendizagem da programação.
Ajuda->Comunicar um Erro...: Use isto para comunicar um problema no KTurtle para os programadores. Estes relatórios podem ser usados para tornar as versões futuras do KTurtle melhores.
Ajuda->Sobre o Kturtle: Aqui você poderá encontrar informações sobre o KTurtle, como os seus autores e a licença em que ele se baseia.
Ajuda->Sobre o KDE: Aqui você poderá encontrar informações sobre o KDE. Se não souber ainda o que é o KDE, este é um local que você não poderá perder.

## A Barra de Ferramentas

Aqui você poderá acessar rapidamente às funções mais usadas. Por padrão, você irá encontrar aqui todos os comandos mais úteis, terminando nos ícones Executar os Comandos e o Parar a Execução.
Você poderá configurar a barra de ferramentas usando o menu Configurações->Configurações Avançadas->Configurar Barra de Ferramentas...

## A Barra de Estado

Na barra de estado você poderá saber o estado do KTurtle. Do lado esquerdo, ela fornece o resultado sobre a última ação. Do lado direito, você encontra a localização atual do cursor (os números da linha e da coluna). No meio da barra de estado é indicado idioma usado para os comandos.

# Capítulo 3. Começando

Neste guia introdutório, iremos assumir que o idioma dos comandos é o Inglês. Você poderá mudar este idioma em Configurações->Configurar o KTurtle..., mais precisamente na seção Idioma. Observe que o idioma que definir para o KTurtle precisa ser o mesmo idioma que usa para digitar os comandos de Logo.

## Primeiros passos no Logo: apresentamos a Tartaruga!

Você já deve ter reparado que a tartaruga está no meio da área de desenho: você está agora prestes a aprender como controlá-la, usando os comandos no editor de código.

## A Tartaruga se Move

Vamos começar colocando a tartaruga para andar. A nossa tartaruga tem 3 tipos de movimentos: (1) pode ir para a frente e para trás, (2) pode virar à esquerda ou à direita e (3) poderá ir diretamente para uma posição da tela. Tente isto, por exemplo:

```
forward 100
turnleft 90
```

Digita ou copie e cole o código no editor e execute-o (usando o Arquivo->Executar Comandos) para ver o resultado. Quando tiver digitado e executado os comandos acima no editor de código, você irá reparar em uma ou mais das seguintes coisas:

- Que — depois de executar os comandos — a tartaruga vai mover-se, desenhar uma linha e depois dar um quarto de volta para a esquerda. Isto acontece porque você usou os comandos forward e turnleft.

- Que a cor do código mudou à medida que o foi digitando; este recurso chama-se realce intuitivo — os diferentes tipos de comandos são realçados de forma diferente. Isto torna a leitura de grandes blocos de código mais fácil.

- Que a tartaruga desenha uma linha preta fina.

Talvez tenha obtido uma mensagem de erro. Isto poderá simplesmente significar duas coisas: você poderá ter cometido um erro ao copiar os comandos, ou você precisa de definir o idioma correto para os comandos de Logo (o que você pode fazer isso escolhendo Configurações->Configurar o KTurtle na seção Idioma).

Você irá da mesma forma perceber que o forward 100 mandou a tartaruga andar em frente, deixando uma linha, e que o turnleft 90 disse à tartaruga para virar 90 graus à esquerda.

Por favor, veja as seguintes referências para o manual para uma explicação completa dos comandos inseridos: forward, backward, turnleft e turnright.

O primeiro exemplo foi muito simples, por isso vamos continuar!

```
canvassize 200,200
canvascolor 0,0,0
pencolor 255,0,0
penwidth 5
clear
go 20,20
direction 135
forward 200
turnleft 135
forward 100
turnleft 135
forward 141
turnleft 135
forward 100
turnleft 45
go 40, 100
```

Mais uma vez, você deverá digitar ou copiar e colar o código para o editor ou abrir o arquivo arrow.logo na pasta Abrir exemplos e executá-lo (usando a opção Arquivo->Executar Comandos) para ver o resultado. Nos próximos exemplos, iremos considerar que você já sabe a mecânica do assunto.

Você já deve ter notado que este segundo exemplo usa muito mais código. Você deve ter visto também um conjunto de comandos novos. Aqui está uma breve explicação de todos os comandos novos:

O canvassize 200,200 configura a largura e a altura da área de desenho para 200 pontos. A largura e a altura são iguais em tamanho, o que significa que a área de desenho deverá ser agora um quadrado.

O canvascolor 0,0,0 deixa a área de desenho em preto. O 0,0,0 é uma combinação RGB onde todos os valores ficam iguais a 0, o que corresponde a preto.

O pencolor 255,0,0 deixa a cor do traço em vermelho. O 255,0,0 é uma combinação RGB em que só o valor do 'vermelho' fica igual a 255 enquanto que os outros (verde e azul) ficam a 0, o que resulta num tom claro de vermelho.

Se não compreender os valores das cores, tente por favor ler o glossário sobre as combinações RGB
O penwidth 5 configura a espessura (ou tamanho) do traço a 5 pontos. A partir de agora, todas as linhas que a tartaruga desenhar irão ter uma espessura de 5 pontos, até que se mude o penwidth para outra coisa qualquer.

O clear limpa a área de desenho, e é tudo o que faz.

O go 20,20 manda a tartaruga ir para um determinado lugar da área de desenho. A contar do canto superior esquerdo, este lugar fica a 20 pontos a contar da esquerda e a 20 pontos da parte superior.

Lembre-se que, ao usar o comando go, a tartaruga não irá desenhar uma linha.

O direction 135 define a direção da tartaruga. O turnleft e o turnright mudam o ângulo da tartaruga a partir da direção atual dela. O direction muda o ângulo da tartaruga a partir do zero e, como tal, não é relativo à posição anterior da tartaruga.

Depois do comando de direção, segue-se um conjunto de comandos forward e turnleft. Estes comandos fazem, de fato, o desenho.

Por fim, é usado outro comando go para mover a tartaruga para o lado.

Certifique-se de seguir as referências. Elas explicam cada comando em mais detalhes.

# Capítulo 4. Referência de Programação em Logo do KTurtle

Esta é a referência para o Logo do KTurtle. Neste capítulo iremos abordar brevemente todos os tipos de instruções diferentes. Depois, serão explicados os comandos, um por um. Em seguida, serão explicados os recipientes, a matemática, as questões e os controladores de execução. Por último, será mostrado como criar os seus próprios comandos com o learn.

## Tipos Diferentes de Instruções

Como em qualquer linguagem, o LOGO possui diferentes tipos de palavras e símbolos. Aqui as diferenças entre os tipos serão brevemente explanadas.

- Comandos

Usando comandos, você diz à tartaruga ou ao KTurtle para fazer algo. Alguns comandos precisam de dados de entrada, enquanto outros fornecem resultados ou dados de saída.

# o 'forward' é um comando que necessita de dados de entrada, neste caso o número 100:

forward 100

- Números

Muito provavelmente você já conhece alguma coisa sobre os números. A forma como eles são usados no KTurtle não é muito diferente do idioma falado ou da matemática.

Temos então os números conhecidos por naturais: 0, 1, 2, 3, 4, 5, etc. Os números negativos: -1, -2, -3, etc. Finalmente, os números decimais ou fracionários, como por exemplo: 0.1, 3.14, 33.3333, -5.05, -1.0.

Os números podem ser usados em cálculos matemáticos e perguntas. Eles também podem ser colocados em recipientes. Os números ficam realçados em azul no editor de código.

- Cadeias de caracteres

Primeiro um exemplo:
print "Olá, sou uma cadeia de caracteres."

Neste exemplo, o print é um comando, enquanto o "Olá, sou uma cadeia de caracteres." é, de fato, uma cadeia de caracteres. Elas começam e terminam com o símbolo "; através destes símbolos, o KTurtle sabe que é uma cadeia de caracteres. As cadeias de caracteres podem ser colocadas em recipientes. Porém, não podem ser usadas em cálculos matemáticos nem em questões. As cadeias de caracteres ficam realçadas em vermelho escuro no editor de código.

- Nomes

Ao usar a linguagem de programação Logo, você pode criar coisas novas. Se criar um programa irá necessitar normalmente de recipientes e, em alguns casos, do learn para criar novos comandos. Ao criar um recipiente ou um novo comando com o learn, você terá que especificar um nome.

Você poderá escolher qualquer nome, desde que ele não tenha já algum significado. Por exemplo, não poderá chamar um recipiente de forward, uma vez que esse nome já é usado para um comando e, assim, já tem um significado.

```
# Aqui o forward é usado como recipiente, mas já tem um significado
# assim irá produzir um erro:
forward = 20

# isto funciona:
forward 20
```

Os nomes só poderão conter letras, números e sublinhados (_). De qualquer forma, devem começar por uma letra.  Por favor, leia a documentação sobre os recipientes e o comando learn para uma melhor explicação e mais exemplos.

- Atribuições

As atribuições são feitas com o símbolo =. Nas linguagens de programação é melhor ler o = simples não como um 'é igual a' mas sim como um 'ficar igual a'. O termo 'é igual a' é mais apropriado para o ==, que é uma pergunta. As atribuições são usadas normalmente por duas razões, (1) para adicionar conteúdo aos recipientes e (2) para modificar o conteúdo de um recipiente. Por exemplo:

x = 10

# o recipiente 'x' contém agora o número 10
W = "A minha idade é: "
# o recipiente W contém agora o texto "A minha idade é: "
# isto imprime o conteúdo dos recipientes 'W' e 'x' na área de desenho
print W + x

Para mais exemplos, veja a seção que explica os recipientes.

- Símbolos Matemáticos

O KTurtle suporta todos os símbolos matemáticos básicos: a adição (+), a substração (-), a multiplicação (*), a divisão (/) e os parênteses ( e ). Para uma explicação completa e mais exemplos, veja a seção de matemática.

- Perguntas

Nós podem fazer perguntas simples onde a respostar será 'true' (verdadeiro) ou 'false' (falso). O uso de perguntas é extensivamente explicado na seção perguntas.

- Palavras de Colagem de Perguntas

As perguntas podem ser coladas juntas com o que se denomina por 'cola das perguntas'. As palavas de colagem são o and (e), o or (ou) e uma palavra especial: a not (não). A utilização da colagem de perguntas é explicada na seção de Colagem de Perguntas.

- Comentários

Comentários são linhas que iniciam com um #. Por exemplo:

# isto é um comentário!
print "isto não é um comentário"
# a linha anterior não é um comentário, mas a próxima é:
# print "isto não é um comentário"

Nós podemos adicionar comentários ao código para nós mesmos ou para que alguém os leia. Comentários são usados para: (1) adicionar uma pequena descrição ao programa, (2) explanar como um pedaço do código funciona se ele for um pouco complicado, e (3) para 'comentar' linhas de código que devem ser (temporariamente) ignoradas (veja a última linha do exemplo). As linhas comentadas ficam realçadas em amarelo escuro no editor de código.

- Comandos

Ao usar os comandos, você diz à tartaruga ou ao KTurtle para fazer algo. Alguns comandos precisam de dados de entrada, enquanto outros fornecem resultados ou dados de saída. Nesta seção iremos explicar todos os comandos que podem ser usados no KTurtle. Observe que todos os comandos incorporados ficam realçados em verde escuro no editor de código, para que possa ajudar a distinguí-los.

## Movendo a tartaruga

Existem vários comandos para mover a tartaruga pela tela.

- forward X (fw X)

O forward move a tartaruga para a frente X pixels. Quando o traço está em baixo, a tartaruga irá deixar um rastro. O forward pode ser abreviado para fw

- backward X (bw X)

O backward move a tartaruga para trás X pixels. Quando o traço está em baixo, a tartaruga irá deixar um rastro. O backward pode ser abreviado para bw.

- turnleft X (tl X)

O turnleft diz à tartaruga para se virar X graus para a esquerda. O turnleft pode ser abreviado para tl.

- turnright X (tr X)

O turnright diz à tartaruga para se virar X graus para a direita. O turnright pode ser abreviado para tr.

- direction X (dir X)

O direction configura a direção da tartaruga para um ângulo de X graus a contar do zero, e isto não é relativo à direção anterior da tartaruga. O direction pode ser abreviado para dir.

- center

O center move a tartaruga para o centro da área de desenho.

- go X,Y

O go manda a tartaruga ir para um determinado local da área de desenho. Este local está a X pixels do lado esquerdo da área de desenho e a Y pixels do topo da área. Lembre-se que, ao usar o comando go, a tartaruga não irá desenhar nenhuma linha.

- gox X

Ao usar o comando gox, a tartaruga irá mover-se X pixels a partir da esquerda da área de desenho, mantendo-se na mesma altura.

- goy Y

Ao usar o comando goy, a tartaruga irá mover-se Y pixels a partir do topo da área de desenho, mantendo-se na mesma distância do lado esquerdo da área de desenho.

## A tartaruga tem um traço

A tartaruga tem um traço e vai desenhando uma linha à medida que a tartaruga se move. Existem alguns comandos para controlar o traço. Nesta seção iremos explicar estes comandos.

- penup (pu)

O penup levanta o traço da área de desenho. Quando o traço está “em cima”, não é desenhada

nenhuma linha à medida que a tartaruga se move. Veja também o pendown. O penup pode ser abreviado para pu.

- pendown (pd)

O pendown pressiona o traço para baixo na área de desenho. Quando o traço está "em baixo", é desenhada uma linha à medida que a tartaruga se move. Veja também o penup. O pendown pode ser abreviado para pd.

- penwidth X (pw X)

O penwidth configura a espessura do traço para X pixels. O penwidth pode ser abreviado para pw.

- pencolor R,G,B (pc R, G, B )

O pencolor configura a cor do traço. O pencolor recebe uma combinação de RGB como parâmetro. O pencolor pode ser abreviado para pc.

## Comandos para controlar a área de desenho

Existem vários comandos para controlar a área de desenho.

- canvassize X,Y (cs X, Y)

Com o comando canvassize você poderá alterar o tamanho da área de desenho. Ele recebe dois parâmetros (X e Y) de entrada, em que o X é a nova largura da área de desenho em pixels, e o Y é a nova altura da mesma área em pixels. O canvassize pode ser abreviado para cs.

- canvascolor R,G,B (cc R, G, B)

O canvascolor define a cor da área de desenho. O canvascolor recebe uma combinação RGB como parâmetro. O canvascolor pode ser abreviado para cc.

- wrapon

Com o comando wrapon você poderá ativar a envolvência para a área de desenho. Por favor veja o glossário para saber o que é a envolvência.

- wrapoff

Com o comando wrapoff você poderá desativar a envolvência para a área de desenho. Isto significa que a tartaruga poderá mover-se para fora da área de desenho e "perder-se". Por favor veja o glossário se quiser saber o que é a envolvência.

## Comandos para limpar

Existem dois comandos para limpar a área de desenho, depois de você ter deixado tudo bagunçado.

- clear (cr)

Com o clear, você poderá limpar todos os desenhos da área de desenho. Todo o resto permanece igual: a posição e o ângulo da tartaruga, a cor da área de trabalho, a visibilidade da tartaruga e o

tamanho da área de desenho. O clear pode ser abreviado para cr.

- reset

O reset limpa tudo de forma mais abrangente que o comando clear. Depois de um comando reset, tudo fica como estava quando você iniciou o KTurtle. A tartaruga é posicionada no meio do tela, a cor da área de desenho é branca e a tartaruga irá desenhar uma linha preta na área de desenho.

## A tartaruga é uma imagem móvel

Muitas pessoas não sabem o que são as imagens móveis ('sprites'), daí uma breve explicação: as imagens móveis são pequenas imagens que podem percorrer o tela (para mais informações, veja o glossário sobre as imagens móveis).

A seguir você irá encontrar uma apresentação completa de todos os comandos que lidam com imagens móveis.

[A versão atual do KTurtle não suporta ainda o uso de imagens móveis além da tartaruga. Nas versões futuras, você poderá mudar a tartaruga para outra coisa que desejar]

- show (ss)

O show torna a tartaruga visível de novo depois de ter ficado escondida. O show pode ser abreviado para ss.

- hide (sh)

O hide esconde a tartaruga. Isto pode ser usado se a tartaruga não couber no seu desenho. O hide pode ser abreviado para sh.

## Será que as tartarugas sabem escrever?

A resposta é: “sim”. A tartaruga sabe escrever e pode escrever tudo o que lhe disser para escrever.

- print X

O comando print é usado para dizer à tartaruga para escrever algo na área de desenho. O print recebe números e texto como parâmetros. Você poderá executar o print para vários parâmetros com o sinal “+”. Veja aqui um pequeno exemplo:

```
ano = 2004
autor = "Ze"
print "O " + autor + " iniciou o projeto do KTurtle em " + ano + " e ainda continua gostando de trabalhar nele!"
```

- fontsize X

O fontsize configura o tamanho da letra que é usado pelo print. O fontsize recebe um parâmetro que deverá ser um número. O tamanho é definido em pixels.

## Um comando que joga os dados para você

Existe um comando que lança os dados para você, que se chama random; ele é muito útil para alguns resultados inesperados.

- random X,Y

O random é um comando que recebe parâmetros e devolve resultados. Como parâmetros são necessários dois números, onde o primeiro define o resultado mínimo (X) e o segundo o máximo (Y). O resultado é um número escolhido aleatoriamente que é maior ou igual ao mínimo e menor ou igual ao máximo. Aqui está um pequeno exemplo:

```
repeat 500 [
  x = random 1,20
  forward x
  turnleft 10 - x
]
```

Com o comando 'random', você poderá adicionar um pouco de confusão ao seu programa.

## Entrada de dados e reação através de janelas

Uma janela poderá pedir alguma alguma reação em especial ou a introdução de determinados dados. O KTurtle tem dois comandos para janelas, nomeadamente o message e o inputwindow

- message X

O comando message recebe uma cadeia de caracteres como entrada. Mostra então uma janela que contém o texto da cadeia de caracteres.

```
ano = 2004
autor = "Ze"
print "O " + autor + " iniciou o projeto do KTurtle em " + ano + " e ainda continua gostando de trabalhar nele!"
```

- inputwindow X

O inputwindow recebe uma cadeia de caracteres como entrada. Mostra uma janela que contém o texto da cadeia de caracteres, tal como acontece no message. Contudo, além disso, também mostra um campo de texto na janela. Através deste campo, o usuário poderá introduzir um número ou uma cadeia de caracteres que poderá ser guardada num recipiente. Por exemplo

```
in = inputwindow "Que idade você tem?"
out = 2003 - in
print "Em 2003, você tinha " + out + " anos em determinado momento."
```

Quando um usuário cancelar a janela ou não inserir nenhuma informação, o recipiente fica vazio.

## Recipientes

Os recipientes são letras ou palavras que podem ser usadas pelo programador para guardar algum número ou algum texto. Os recipientes que contém um número chamam-se variáveis, enquanto que os que contém texto chamam-se cadeias de caracteres. Os recipientes que não são usados não contém nada. Por exemplo, um:

```
print N
```

Isto não irá imprimir nada. Se tentar fazer operações matemáticas com recipientes vazios, irá obter erros.

- ## Variáveis

Variáveis são recipientes de números, vamos começar com um exemplo:

```
x = 3
print x
```

Na primeira linha, a letra x passou a ser uma variável (um recipiente de números). Como você pode ver, o valor da variável x passou para 3. Na segunda linha, o valor é impresso.
Lembre-se que, se quisermos imprimir um “x”, então devemos escrever
print "x"

Isso foi fácil, mas agora há um exemplo um pouco mais difícil:

```
A = 2004
B = 25
C = A + B

# o próximo comando imprime "2029"
print C
backward 30
# o próximo comando imprime "2004 mais 25"
print A + " mais " + B
backward 30
# o próximo comando imprime "1979"
print A - B
```

Nas duas primeiras linhas, as variáveis A e B são configuradas como sendo iguais a 2004 e 25. Na terceira linha, a variável C fica igual a A + B, o que dá 2029. O resto do exemplo consiste em 3 comandos print com backward 30 no meio. O backward 30 está lá para garantir que cada resultado fica numa linha diferente. Neste exemplo, você vê também que as variáveis podem ser usadas nos cálculos matemáticos.

- ## Recipientes de texto (cadeias de caracteres)

No código do programa, o texto normal é iniciado e termina normalmente com aspas. Como já foi visto:
print "Olá programador!"

O texto fica delimitado com as aspas. Estes pedaços de texto normal são chamados então de cadeias de caracteres. As cadeias de caracteres são bastante parecidas com as variáveis. A maior diferença é que as cadeias de caracteres não podem ser usadas em cálculos matemáticos e perguntas. Um exemplo da utilização das cadeias de caracteres:

```
x = "Olá "
nome = inputwindow "por favor insira o seu nome..."
print x + nome + ", como está?
```

Na segunda linha, a cadeia de caracteres x fica igual a “Olá ”. Na segunda linha, a cadeia de

caracteres nome é configurada como o resultado do comando inputwindow. Na terceira linha, o programa imprime uma composição de três cadeias de caracteres na área de desenho.

Este programa pede para inserir o seu nome. Quando você, por exemplo, inserir o nome “José”, o programa irá imprimir “Olá José, como está?”. Lembre-se que o sinal de mais (+) é o único símbolo matemático que você poderá usar com as cadeias de caracteres.

## Será que a Tartaruga sabe fazer contas?

Sim, o KTurtle sabe fazer contas para você. Você poderá somar (+), subtrair (-), multiplicar (*) e dividir (/). Aqui está um exemplo no qual iremos usar todas as operações:

```
a = 20 - 5
b = 15 * 2
c = 30 / 30
d = 1 + 1
print "a: "+a+", b: "+b+", c: "+c+", d: "+d
```

Será que você sabe o valor de 'a', 'b', 'c' e 'd'? Repare por favor no uso do símbolo = de atribuição. Se você somente queria fazer um cálculo simples, você poderá fazer algo semelhante a isto:

```
print 2004-12
```

Agora, um exemplo com parênteses:

```
print ( ( 20 - 5 ) * 2 / 30 ) + 1
```

O que estiver entre parênteses será calculado em primeiro lugar. Neste exemplo, o 20-5 será calculado, depois será multiplicado por 2, dividido por 30 e depois é adicionado 1 (o que dá 2).

## Fazendo perguntas, obtendo respostas...

O if e o while são controladores de execução que iremos discutir na próxima seção. Nesta seção iremos usar o comando if para explicar as perguntas.

Perguntas
Um exemplo simples de pergunta:

```
x = 6
if x > 5 [
  print "olá"
]
```

Neste exemplo, a pergunta é x > 5, se a resposta a esta pergunta for “true” (verdadeira), o código entre colchetes será executado. As perguntas são uma parte importante da programação e são usadas normalmente em conjunto com os controladores de execução, como o if. Todos os números e variáveis (recipientes de números) poderão ser comparados uns com os outros nas perguntas.

| *Teste* | *Processamento* |
|---|---|
| a == b | é igual a  a resposta é “true” (verdadeira) se o a for igual ao b |
| a != b | é diferente de  a resposta é “true” (verdadeira) se o a não for igual ao b |
| a > b | maior que  a resposta é “true” (verdadeira) se o a for maior que o b |

| *Teste* | *Processamento* |
|---|---|
| a < b | menor que  a resposta é “true” (verdadeira) se o a for menor que o b |
| a >= b | maior ou igual a  a resposta é “true” (verdadeira) se o a for maior ou igual ao b |
| a <= b | menor ou igual a  a resposta é “true” (verdadeira) se o a for menor ou igual ao b |

Tabela 4.1. Tipos de perguntas

As perguntas ficam realçadas em azul claro no editor de código.

## Colagem de Perguntas

As perguntas também podem ser coladas umas às outras com “palavras de colagem de perguntas” onde, desta forma, algumas perguntas tornam-se uma pergunta maior.

```
a = 1
b = 5
if (a < 5) and (b == 5) [
  print "olá"
]
```

Neste exemplo, a palavra de colagem and é usada para colar 2 perguntas (a < 5, b == 5) juntas. Se um lado do and der uma resposta “false” (falsa), a pergunta toda irá responder “false”, porque, com a palavra de colagem and, ambos os lados precisam ser “true” para que a resposta seja “true” (verdadeira). Por favor não se esqueça de usar os parênteses ao redor das perguntas!

| *Conector* | *Resultado do Teste* |
|---|---|
| and | ambos os lados têm que ser “true” (verdadeiros) para a resposta ser “true” (verdadeira) |
| or | se um dos lados for “true” (verdadeiros) a resposta é “true” (verdadeira) |
| not | Caso especial: só funciona com uma pergunta! Muda o 'true' para 'false' e o 'false' para 'true'. |

Tabela 4.2. Palavras de colagem de perguntas

As palavras de colagem ficam realçadas em púrpura no editor de código.

- and

Quando são coladas duas perguntas juntas com o and, ambos os lados do and terão que ser 'true' para que o resultado também seja 'true' (verdadeiro). Por exemplo:

```
a = 1
b = 5
if ((a < 10) and (b == 5)) and (a < b) [
  print "olá"
]
```

Nestes exemplo, você poderá ver uma pergunta colada a outra pergunta também colada.

- or

Se uma das duas perguntas coladas juntas com o or for 'true' (verdadeira), o resultado será também 'true'. Por exemplo:

```
a = 1
```

```
b = 5
if ((a < 10) or (b == 10)) or (a == 0) [
  print "olá"
]
```

Neste exemplo, irá ver uma pergunta colada a outra pergunta, que também está colada.

- not

O not é uma palavra de colagem especial porque somente funciona com uma pergunta de cada vez. not muda 'true' para 'false' e 'false' para 'true'. Um exemplo:

```
a = 1
b = 5
if not ((a < 10) and (b == 5)) [
  print "olá"
]
else
[
  print "adeus ;-)"
]
```

Neste exemplo a pergunta colada é 'true' ainda que o not mude-a para 'false'. Assim, no final, "adeus ;-)" é impresso na área de desenho.

## Controlando a execução

Os controladores de execução permitem-lhe — como o nome deles indica — controlar a execução. Os comandos de controle de execução são realçados em verde escuro e negrito. Os colchetes são frequentemente usados juntamente com os controladores de execução e eles são realçados em verde claro.

- wait X

Se você já tentou programar um pouco no KTurtle, você já poderá ter reparado que a tartaruga pode ser bastante rápida desenhando. Este comando faz a tartaruga esperar X segundos.

```
repeat 36 [
  forward 5
  turnright 10
  wait 0.5
]
```

Este código irá desenhar uma circunferência, mas a tartaruga irá esperar meio segundo a cada passo. Isto dá a noção de uma tartaruga vagarosa.

- if pergunta [ ... ]

O código que é colocado no “...” só será executado se (if) a resposta à pergunta for “true” (verdadeira). Por favor leia, para obter mais informações sobre perguntas, a seção de perguntas.

```
x = 6
if x > 5 [
```

```
  print "O x é maior que cinco!"
]
```

Na primeira linha, o x é inicializado com 6. Na segunda linha, a pergunta x > 5 é feita. Uma vez que a resposta a esta pergunta é verdadeira, o controlador de execução if irá permitir que o código entre colchetes seja executado

- while pergunta [ ... ]

O controlador de execução while é um pouco parecido com o if. A diferença é que o while fica repetindo (em ciclo) o código entre os colchetes até que a resposta à pergunta seja "false".

```
x = 1
while x < 5 [
  forward 10
  wait 1
  x = x + 1
]
```

Na primeira linha o x é inicializado com 1. Na segunda linha a pergunta x < 5 é feita. Uma vez que a resposta é "verdadeiro" o controlador de execução while inicia a execução do código entre colchetes até que a resposta à pergunta seja "falso". Neste caso o código entre colchetes será executado 4 vezes, porque cada vez que a quinta linha é executada o x é aumentado de 1.

- if pergunta [ ... ] else [ ... ]

Se não, em outras palavras: "else". O else pode ser usado em adição ao controlador de execução if. O código entre colchetes após o else só é executado se a resposta à pergunta feita é respondida como "falso". [O else não esta funcionando na versão 0.6]

- reset

```
x = 4
if x > 5 [
  print "x é maior que cinco!"
]
else
[
  print "x é menor que seis!"
]
```

A pergunta feita é se x é maior que 5. Uma vez que x é inicializado com 4 na primeira linha a resposta à questão é "falso". Isto significa que o código entre colchetes após o else será executado.

- for ponto inicial a ponto final [ ... ]

O ciclo for é um "ciclo de contagem", ou seja, faz um contador para você.

```
for x = 1 to 10 [
  print x * 7
  forward 15
]
```

Cada vez que o código entre parênteses é executado, o x é incrementado de uma unidade, até que o

valor do x chegue a 10. O código entre parênteses imprime o valor de x multiplicado por 7. Depois de este programa terminar a sua execução, você irá ver a tabuada dos 7 na área de desenho.

## Crie os seus próprios comandos com o “learn”

O learn é um comando muito especial, porque ele é usado para criar seus próprios comandos. O comando que você criar pode ter valores de entrada e retornar valores de saída. Vamos dar uma olhada em como um novo comando é criado:

```
learn circulo x [
  repeat 36 [
    forward x
    turnleft 10
  ]
]
```

O novo comando é chamado circulo. circulo recebe um valore de entrada, um número, para configurar o tamanho do círculo. circulo não retorna nenhum valor de saída. O comando circulo pode agora ser usado como um comando normal no resto do código. Veja este exemplo:

```
learn circulo (x) [
  repeat 36 [
    forward x
    turnleft 10
  ]
]

go 30,30
circulo 20

go 40,40
circulo 50
```

No próximo exemplo, um comando com valor de retorno é criado.

```
reset

learn multPorSi n [
  r = n * 1
  r = n * n
  return r
]
i = inputwindow "Por favor insira um número e pressione OK"
print i + " multiplicado por ele mesmo é: " + multPorSi i
```

Neste exemplo um novo comando chamado multPorSi é criado. A entrada deste comando é multiplicada por ela mesmo e então retornada, usando o comando return. O comando return é a maneira de retornar um valor a partir de uma função que você criou.

# Capítulo 5. Glossário

Neste capítulo, você irá obter uma explicação para a maioria das palavras “pouco comuns” que são usadas no manual.

- graus

Os graus são uma unidade para medir ângulos ou voltas. Uma volta completa corresponde a 360 graus, uma meia-volta corresponde a 180 graus e um quarto-de-volta a 90 graus. Os comandos turnleft, turnright e direction necessitam de um parâmetro em graus.

- parâmetros e resultado dos comandos

Alguns comandos recebem parâmetros, outros devolvem resultados, outros fazem ambas as coisas e finalmente existem outros que não fazem nenhuma delas.
Alguns exemplos de comandos que só recebem parâmetros são:
forward 50
pencolor 255,0,0
print "olá!"

O comando forward recebe o 50 como parâmetro, porque o forward precisa deste parâmetro para saber quantos pontos deverá andar em frente. O pencolor recebe um parâmetro e o print recebe uma cadeia de caracteres com parâmetro. Lembre-se que o parâmetro também poderá ser um recipiente. O próximo exemplo ilustra isto:
x = 50
print x
texto = "olá!"
print texto

Agora alguns exemplos de comandos que devolvam resultados:
x = inputwindow "por favor digite algo e pressione OK... obrigado!"
r = random 1,100

O comando inputwindow recebe um texto como parâmetro e devolve o número ou o texto que é introduzido. Como poderá ver, o resultado do inputwindow é guardado no recipiente x. O comando random também devolve um resultado. Neste caso, devolve um número entre 1 e 100. O resultado do random é de novo guardado num recipiente, chamado r. Lembre-se que os recipientes x e r não são usados no código de exemplo acima.

Também existem alguns comandos que não precisam de parâmetros nem devolvem nada. Alguns exemplos:
clear
penup
wrapon
hide

- realce intuitivo

Este é um recurso do KTurtle que torna a codificação ainda mais simples. Com o realce intuitivo, o código que você escrever ganha uma cor que indica qual tipo de código é. Na próxima lista, você irá encontrar os diferentes tipos de código e a cor que obtém no editor de código.

| ***Código*** | ***Côr*** | ***Descrição ou exemplos*** |
|---|---|---|
| comandos normais | verde escuro | forward, backward, turnleft, turnright, direction, center, go, gox, goy, penup, pendown, penwidth, pencolor, canvasize, canvascolor, wrapon, wrapoff, clear, reset, show, hide, print, fontsize, random, message, inputwindow |
| controladores de execução | preto (negrito) | wait, if, else, while, repeat, for |

| *Código* | *Côr* | *Descrição ou exemplos* |
|---|---|---|
| comentários | amarelo escuro | As linhas que estão comentadas começam por caracteres de comentário (#); estas linhas são ignoradas quando o código é executado. Os comentários permitem ao programador explicar um pouco do seu código ou podem ser usadas para evitar temporariamente que um pedaço de código seja executado. |
| colchetes [, ] | verde claro (negrito) | Os colchetes são usados para agrupar pedaços de código. Os colchetes são usados normalmente com os controladores de execução. |
| learn | verde claro (negrito) | O comando learn é usado para criar comandos novos. |
| números | azul | Números, bem não temos muita coisa para falar sobre eles. |
| texto | vermelho escuro | Também não há muito a dizer sobre o texto nas cadeias de caracteres, a não ser que começam e terminam com aspas ("). |
| caracteres matemáticos | cinza | Estes são os caracteres matemáticos: +, -, *, /, (, e ). |
| caracteres das perguntas | azul (negrito) | |
| palavras de colagem das perguntas | cor de rosa | 'and', 'or' e 'not' |
| texto normal | preto | |

Tabela 5.1. Os diferentes tipos de código e a sua cor de realce

- pontos

Um ponto é um ponto na tela. Se você olhar muito de perto para o que vê na tela do seu monitor, irá constatar que ela usa pontos. Todas as imagens da tela são criadas com estes pontos. Um ponto é a menor coisa que poderá ser desenhada na tela.

Existem vários comandos que precisam de uma quantidade de pontos como parâmetro, e são: o forward, backward, go, gox, goy, canvassize e o penwidth.

- Combinações de RGB (códigos de cores)

As combinações de RGB são usadas para descrever cores. O “R” vem de “red” (vermelho), o “G” de “green” (verde) e o “B” de “blue” (azul). Um exemplo de uma combinação RGB é o 255,0,0, onde o valor da componente vermelha é 255 e as outras são 0, o que resulta num tom claro de vermelho. Cada valor de uma combinação RGB terá que estar no intervalo entre 0 e 255. Aqui está uma lista com as cores mais usadas:

| *RGB* | *Côr* |
|---|---|
| 0,0,0 | preto |
| 255,255,255 | branco |
| 255,0,0 | vermelho |
| 150,0,0 | vermelho escuro |
| 0,255,0 | verde |
| 0,0,255 | azul |
| 0,255,255 | azul claro |
| 255,0,255 | cor de rosa |
| 255,255,0 | amarelo |

Tabela 5.2. Combinações RGB mais usadas

Para descobrir facilmente as combinações RGB de uma cor, você deverá experimentar o extrator de cores! Você poderá encontrá-lo aqui: Ferramentas->Extrator de Cor.

Dois comandos necessitam de uma combinação RGB como parâmetro, e são eles: o canvascolor e o pencolor.

- imagem móvel

Uma imagem móvel é uma pequena imagem que pode ser movida pela tela. A nossa tartaruga é uma imagem móvel, por exemplo.

Nota: com esta versão do KTurtle, a imagem móvel não consegue ser alterada de uma tartaruga para outra coisa. As versões futuras do KTurtle serão capazes de fazer isso.

- envolvência

A envolvência é o que acontece quando a tartaruga desenha algo que é muito grande para caber na área de desenho e a envolvência está ativa. Quando a tartaruga passa para fora de um extremo da área de desenho ele vai passar para o extremo imediatamente oposto para que possa continuar o seu movimento. Deste modo, a tartaruga irá estar sempre na tela, enquanto se move. A envolvência poderá ser ativada e desativada com os comandos wrapon e wrapoff. Quando o KTurtle inicia, a envolvência está ativa por padrão.

# Capítulo 6. Guia do Tradutor do Kturtle

Como você sabe, um dos grandes recursos da linguagem Logo é que os comandos de Logo podem ser traduzidos para o seu próprio idioma. Isto facilita a uma criança entender os comandos. Para habilitar um idioma novo, existem três arquivos a traduzir: em primeiro lugar, o arquivo de palavras-chave (ou comandos), depois o arquivo 'logo-highlight-style' e finalmente os exemplos.

## Criando um Dicionário para Guardar os Arquivos Traduzidos

Primeiro, você precisa criar uma pasta para guardar os arquivos traduzidos. Crie uma pasta chamada kde-i18n/código/data/kdeedu/kturtle/ na sua pasta de CVS do KDE, onde o código é o código do seu país (o código ISO de 2 ou 4 letras).

Copie o arquivo Makefile.am de kdeedu/kturtle/data/ para esta pasta. Abra-o com o seu editor de texto favorito, substitua todas as instâncias de “en_US” no arquivo pelo seu código de país (o que é usado acima) e grave o arquivo.

## Como Traduzir as Palavras-Chave (Comandos) do Logo

Copie o arquivo logokeywords.en_US.xml, mudando o seu nome para logokeywords.codigo.xml em que o codigo é o código do seu país (para o Brasil, é o br).

Traduza para o seu próprio idioma o conteúdo da marca <keyword>, isto é a informação entre o <keyword> e o </keyword> sempre que possível e o conteúdo da marca <alias>, isto é a informação entre o <alias> e o </alias>. Estas informações estão relacionadas, uma vez que o conteúdo do 'alias' é um nome alternativo ou atalho para a palavra-chave.

Por exemplo, o “while” traduz-se em português para: <keyword>enquanto</keyword>.

Por favor não traduza mais nada e não traduza as palavras em inglês no <command name="palavra_em_ingles">. Estas terão que permanecer em Inglês.

Salve o seu arquivo como UTF-8 (no Kate, use o Salvar Como... e mude para UTF-8 na lista à direita do nome do arquivo).

Envie o seu arquivo por CVS (adicione o nome do seu arquivo ao Makefile.am) ou envie-o para a Anne-Marie. Em caso de qualquer dúvida, por favor contacte a Anne-Marie Mahfouf (annemarie.mahfouf AT free.fr) para mais informações.

## Como Traduzir os Arquivos de Realce de Sintaxe

Copie o arquivo logohighlightstyle.en_US.xml, mudando o nome dele para logohighlightstyle.codigo.xml em que o codigo é o código de 2 ou 4 letras ISO do seu país (no caso do Brasil, é o 'br').

A linha 4 do arquivo, tem um <language name="en_US"> ..., o qual você deverá alterar para o código ISO do seu idioma ("pt_BR", para o Português do Brasil).

Traduza para o seu próprio idioma o conteúdo da marca <item> i.e a informação entre o <item> e o </item>. Este conteúdo deverá ter uma correspondência ao arquivo logokeyword. Por exemplo, traduza o "while" para: <item> enquanto </item> e deixe os espaços tal como estão (um no início e um no fim). Por favor não traduza mais nada.

Salve o seu arquivo como UTF-8 (no Kate, use o Salvar Como... e mude para UTF-8 na lista à direita do nome do arquivo).

Envie o seu arquivo por CVS (adicione o nome do seu arquivo ao Makefile.am) ou envie-o para a Anne-Marie. Em caso de qualquer dúvida, por favor contacte a Anne-Marie Mahfouf (annemarie.mahfouf AT free.fr) para mais informações.

## Como Traduzir os Exemplos

Copie os exemplos em Inglês da pasta kdeedu/kturtle/data/ e mude os nomes dos arquivos de acordo com a tradução para o seu idioma: isto permitirá aos usuários perceberem rápida e facilmente o objetivo do exemplo.

Traduza as palavras-chave nos exemplos, usando as do logokeywords.xml para o seu idioma. O arquivo de palavras-chave deverá estar terminado, em primeiro lugar, antes de traduzir os exemplos.

Salve o seu arquivo como utf-8 (no Kate, use o Salvar Como... e mude para UTF-8 na lista à direita do nome do arquivo).

Envie a sua pasta (adicione um Makefile.am dentro dela) ou envie-a para a Anne-Marie. Em caso de alguma dúvida, por favor contacte a Anne-Marie Mahfouf, (annemarie.mahfouf AT free.fr) para mais informações.

Finalmente, se você quiser, poderá adicionar os seus próprios exemplos nesta pasta.

# Capítulo 7. Créditos e Licença

Kturtle: Direitos autorais do programa 2003-2004 Cies Breijs (cies AT kde DOT nl)

Contribuições: Ajuda na codificação, componente de edição: Anne-Marie Mahfouf (annma AT kde DOT org)

Autor do “WSBASIC” (wsbasic.sourceforge.net), a base para o interpretador do KTurtle: Walter Schreppers (Walter DOT Schreppers AT ua DOT ac DOT be)
Arquivo de Dados em Alemão: Matthias Meßmer (bmlmessmer AT web DOT de)
Arquivo de Dados em Alemão: Burkhard Lück (lueck AT hube-lueck DOT de)
Arquivo de Dados em Sueco: Stefan Asserhäll (stefan DOT asserhal AT telia DOT com)
Arquivos de Dados em Esloveno: Jure Repinc (jlp AT holodeck1.com)
Arquivos de Dados em Sérvio (Cirílico e Latim): Chusslove Illich (caslav.ilic AT gmx.net)
Arquivos de Dados em Italiano: Pino Toscano (toscano.pino AT tiscali.it)
Arquivos de Dados em Inglês Britânico: Andy Potter (A.J.Potter AT rhul.ac.uk)
Arquivo de Dados em Sueco: Stefan Asserhäll (stefan DOT asserhal AT telia DOT com)
Suporte do processador de cirílico: Albert Astals Cid (astals11 AT terra.es)
Direitos autorais da documentação 2004
Cies Briej (cies AT showroommama DOT nl)
Anne-Marie Mahfouf (annma AT kde DOT org)
Algumas alterações de correção do texto por Philip Rodrigues (phil AT kde.org)
Ajuda de tradução atualizada e algumas mudanças de verificação editorial de Andrew Coles (andrew_coles AT yahoo DOT co DOT uk)
Tradução de Marcus Gama(marcus_gama AT uol.com.br)

Esta documentação é licenciada sob os termos da Licença de Documentação Livre GNU.
Este programa é licenciado sob os termos da Licença Pública Geral GNU.

# Apêndice A. Instalação

Como obter o KTurtle
KTurtle faz parte do projeto KDE http://www.kde.org/.
KTurtle pode ser encontrado no pacote kdeedu no site FTP principal do projeto KDE ftp://ftp.kde.org/pub/kde/.

Compilação e Instalação
Para compilar e instalar o KTurtle sem seu sistema, digite o sequinte no diretório base da distribuição de KTurtle:
% ./configure
% make
% make install

Uma vez que o KTurtle usa o autoconf e o automake você não deve ter problemas em compilá-lo. Se você tiver problemas por favor reporte-os às lista de correio do KDE.